JOURNAL OF COMPUTER AND SYSTEM SCIENCES 47, 437–458 (1993)

Output Sensitive and Dynamic Constructions of Higher Order Voronoi Diagrams and Levels in Arrangements

KETAN MULMULEY*

The University of Chicago, Chicago, Illinois

Received August 21, 1990; revised February 6, 1992

We give efficient, output sensitive algorithms to construct Voronoi diagrams of order one to k of a given collection of sites in R^d and levels of order one to k in a nonredundant arrangement of hyperplanes in R^d (an arrangement is called nonredundant if every hyperplane in it supports the convex polytope of the arrangement surrounding the origin). We also give efficient dynamic algorithms for the same problems that allow the user to add or delete an object—the object being a site in the case of Voronoi diagrams and a nonredundant hyperplane in the case of levels. © 1993 Academic Press, Inc.

1. INTRODUCTION

A kth order Voronoi diagram [9] of a given collection of sites in R^d is a partition of R^d into a collection of convex regions, where each region R is labeled with a set of k sites which are precisely the k closest sites to any point in R. If we are interested in the farthest k sites instead, we obtain a diagram that is a dual of the kth order Voronoi diagram. Because dualization of our algorithms is easy, we shall not be concerned with the dual diagram any more. In this paper we consider the problem of constructing Voronoi diagrams of order one to k in arbitrary dimension. For dimension two, this problem has been well studied. In [14], Lee gave an $O(k^2n \log n)$ algorithm for constructing planar Voronoi diagrams of order one to k. This time was improved to $O(k^2n + n \log n)$ in [1]. A simple randomized algorithm with the same bound on the expected running time was given in [16]. As the size (i.e., the total number of faces) of the first k planar Voronoi diagrams is of the order of k^2n [14], for $k \leqslant n/2$, the latter running time is optimal.

In this paper we give a deterministic, output sensitive algorithm for constructing Voronoi diagrams in R^d of order one to k, for $d > 2$. Output sensitivity is somewhat redundant in the planar case, because the size of the first k planar Voronoi diagrams is always of the order of k^2n. In higher dimensions, however, this becomes desirable, because the size can range anywhere from $\Omega(k^dn)$ to $O(k^{\lceil (d+1)/2 \rceil}n^{\lfloor (d+1)/2 \rfloor})$. The upper bound on the size follows from [5]. We conjecture that $\Omega(k^dn)$ is a

* Supported by NSF Grant CCR 8906799 and Packard Fellowship. A part of this paper has appeared in the "Proceedings of the 22nd ACM Symposium on Theory of Computing, 1990."

0022-0000/93 $5.00

571/47/3-5

lower bound, but the proof seems elusive. Our algorithm runs in $O(s \log n + k^d n^2)$ time, for $d > 2$, where s is the actual output size of the diagrams (i.e., the total number of faces in it). The first term in the running time estimate is clearly optimal up to a log factor. For k that is not too large, the second term is substantially smaller than $O(k^{\lceil (d+1)/2 \rceil} n^{\lfloor (d+1)/2 \rfloor})$, the worst case size. The k^d factor in the overhead term $k^d n^2$ seems inevitable because, as we said earlier, $\Omega(k^d n)$ is probably the lower bound on the size of the first k Voronoi diagrams. Whether the factor n^2 in the overhead can be replaced by $n \log n$ is open at present, even when $k = 1$, which is the case of ordinary, first-order Voronoi diagrams [18]. A randomized algorithm for constructing the first k Voronoi diagrams, whose expected running time matches the worst case complexity $O(k^{\lceil (d+1)/2 \rceil} n^{\lfloor (d+1)/2 \rfloor})$, for $d > 2$, was given in [16]. Actually, this algorithm has a better expected running time than the worst case bound stated above. The precise bound is in terms of a certain θ series, that can be associated with an arbitrary arrangement of hyperplanes.

We do not know if the *exact* kth order Voronoi diagram can be constructed in a output sensitive fashion, i.e., in time that is linear in the size of the kth-order Voronoi diagram up to a polylog factor. In the planar case, output sensitivity is redundant because the size of the kth Voronoi diagrams is always of the order of $k(n-k)$ [14]. In this case, $O(k(n-k)\sqrt{n} \log n)$ [8] and (randomized) $O(n^{1+\varepsilon} k)$ [4] algorithms are known, but an $O(k(n-k)\,\text{polylog}(n))$ algorithm is not yet known.

Actually, the problem of constructing a Voronoi diagram of order k can be reduced to the problem of constructing the so-called k-level in a nonredundant arrangement of hyperplanes in R^{d+1} [9]. An arrangement is called nonredundant if every hyperplane in the arrangement bounds the convex polytope surrounding the origin. A kth level in an arrangement of hyperplanes in R^d is, crudely speaking, a polyhedral surface, such that each point on this surface is separated by roughly $(k-1)$ hyperplanes from the origin. It follows from this definition that the first level is nothing but the convex polytope surrounding the origin. Thus a k-level in an arrangement is a generalization of the extensively studied notion of a convex polytope. Levels are very important in computational geometry because, besides higher order Voronoi diagrams, several other well-known problems regarding k-sets and half space range queries can be translated to problems concerning levels.

In this paper, we actually give an algorithm for a more general problem of constructing levels of order one to k in a nonredundant arrangement of hyperplanes in R^d. The running time of this algorithm is $O(s \log n + k^{d-1} n^2)$ time, where s is the output size of these levels. It is of interest to consider the special case $k = 1$, which corresponds to the construction of the convex polytope surrounding the origin in an arrangement of hyperplanes. (Note that, if the arrangement were redundant, the hyperplanes which do not bound the convex polytope surrounding the origin can always be discarded in $O(n^2)$ time [15], thereby enforcing our assumption.) Substituting $k = 1$, we obtain the already known result, due to Seidel [18], which states that the facial lattice of a convex polytope can be constructed in $O(s \log n + n^2)$ time, where s is the output size of this lattice. In this way, our result generalizes Seidel's result from $k = 1$ to arbitrary k.

This paper also generalizes two other noteworthy results. These generalizations are crucial ingredients in the final algorithm for constructing levels. The first of these is an elementary but a fundamental result, which states that a (non-degenerate) linear function achieves a unique minimum, if any, on a convex polytope (a minimum might not be attained if the polytope is unbounded). Since a level generalizes the concept of a convex polytope, it is natural ask how many minima a linear function can attain on a level. Obviously there can be several local minima on a level, because levels are not convex, but rather "star shaped." But, just as a convex polytope can have only one minimum even if its size can be as large as $O(n^{\lceil d/2 \rceil})$, it is natural conjecture that, even if the size of the first k levels can be as large as $O(k^{\lfloor d/2 \rfloor} n^{\lceil d/2 \rceil})$, the number of local minima on these levels should be much smaller. In fact, carrying the analogy with convex polytopes even further, one can even expect the bound on the number of these local minima to be independent of n. In this paper we prove such a bound: it is $O(k^d)$.

In [15], Megiddo gave an $O(n)$ time algorithm to determine the minimum of a linear function, say z, on a convex polytope, where n is the number of given half spaces. Since we have proved that there can only be a "few" local z-minima on a level, the question naturally arises as to how fast we can determine these minima. In this paper, we generalize Megiddo's result to levels. We give an algorithm that, given the $(k-1)$th level in an arrangement of hyperplanes, can construct the local z-minima on the kth level in $O(w(k)\, n + s)$ time, where $w(k)$ is the number of new local z-minima on the kth level, and s is the (input) size of the $(k-1)$th level. If we put $k=1$, then $w(1)=1$ and $s=0$ (as the zeroth level is empty) and we recover Megiddo's result. Actually our algoritm uses Megiddo's linear programming algorithm as a subroutine. It is possible to substitute for Megiddo's algorithm any other linear time algorithm such as the one in [6]. This is actually preferable, because the running time estimate on Megiddo's algorithm has a double exponential constant factor inside the "big-oh" notation.

Finally, we also give dynamic algorithms for the two problems considered above, namely the construction of Voronoi diagrams of order one to k in R^d and the construction of levels of order one to k in R^d. In the case of Voronoi diagrams, the dynamic algorithm maintains Voronoi diagrams of order one to k of the "current" set A of sites in R^d. The user is allowed to add or delete a site from A dynamically. The expected running time of our dynamic Voronoi diagram algorithm on a *random* update sequence is $O(k^2 n + n \log n)$, for $d=2$, and $O(k^{\lceil (d+1)/2 \rceil + 1} |A|^{\lfloor (d+1)/2 \rfloor} \log n)$, for $d>2$. If only additions were allowed (the semi-dynamic case) then the sharper bound $O(k^{\lceil (d+1)/2 \rceil} |A|^{\lfloor (d+1)/2 \rfloor})$, for $d>2$, holds. This later bound coincides with the sharp upper bound on the size of Voronoi diagrams of order one to k in R^d.

We must first explain what is meant by a random update sequence and why it is essential to consider such sequences. It is not possible to guarantee good performance on every update sequence for the following simple reason. For the sake of concreteness consider the case of planar first-order Voronoi diagrams. In this case an adversary can choose n sites in R^2 and a sequence of their additions in such a

way that the total structural change in the underlying Voronoi diagram during this sequence of additions is $\Omega(n^2)$. However, such pathological sequences arise rarely. In fact, it can be easily shown that the expected structural change during a random sequence of additions is $O(n)$ [11]. A sequence of additions involving a given set N of sites is called random if it is chosen from the uniform distribution on the set of all addition sequences over N. This notion of a random addition sequence can be easily generalized as follows to the notion of a random update sequence, wherein deletions are allowed. We shall use the *communist* model [17]. Given an update sequence $\bar{u}$, define its signature $\delta = \delta(\bar{u})$ to be the string of $+$ and $-$ symbols such that: the ith symbol in $\delta(u)$ is $+$ (resp. $-$) iff the ith update in $\bar{u}$ is addition (resp. deletion). We say that $\bar{u}$ is an (N, δ)-sequence, where N is the set of objects involved in $\bar{u}$. We say that $\bar{u}$ is a *random* (N, δ)-sequence, if it is chosen from a uniform distribution on all valid (N, δ)-sequences. If we were to stipulate, for the sake of simplicity, that an object, once deleted. during $\bar{u}$, cannot be added again (but we do not really need this assumption), then another way to think about a random (N, δ)-sequence would be: We read the signature δ from the left to right. If the symbol is $+$, we randomly choose an unadded object from N and add it. If the symbol is $-$, we randomly delete one of the earlier added objects. A random $(N, +^n)$-sequence is thus the usual random sequence of additions. In this model an adversary can choose the spatial distribution of the objects in N as well as the signature δ any way he likes. He must, however, treat all objects in N equally as far as the relative order of their additions and deletions is concerned. In that sense, the adversary is a *communist*: he treats all objects (people) equally regardless of their spatial positions.

The case $k = 1$ of our algorithm corresponds to dynamic maintenance of ordinary, first order, Voronoi diagrams.

Because the construction of higher order Voronoi diagrams is reducible to the construction of k-levels, we shall mainly confine ourselves to levels in the rest of this paper. In Section 2, we give some basic definitions. In Section 3 we prove the above mentioned bound on the number of local minima of a linear function on k-levels. Section 4 shows how these minima can be determined efficiently. Section 5 gives an efficient algorithm for constructing the kth level $L_k(N)$ in an arrangement of hyperplanes. Section 6 shows how these results can be put together to obtain an efficient, output sensitive algorithm for constructing the first k levels. Finally, Section 7 gives dynamic algorithms for the maintenance of higher order Voronoi diagrams and levels.

2. Basic Definitions

Let $A(N)$ be an arrangement of a set N of n hyperplanes in R^d. Throughout this paper, we are going to assume that the hyperplanes in N are in general position; this assumption can be readily removed using the standard perturbation arguments. Fix an origin (base point) o in R^d. For any hyperplane H, we shall denote by H^- the

open half space bounded by H that contains the origin o. H^+ will denote the other open half space. We say that a hyperplane H separates a point x from the origin o, if $x \in H^+$. For a point $p \in R^d$, define the level of p to be the number of hyperplanes in the arrangement which separate p from o. The level of any (relatively open) j-face $f \in A(N)$ is defined to be the level of any point in that face; it is easy to see that this definition is consistent. Let $C_k(N)$ denote the subcomplex of $A(N)$ consisting of those faces having level $\leqslant k-1$. $C_k(N)$ will be called the k-complex in $A(N)$ w.r.t. o; if the origin (base point) o is clear from the context we shall not mention it. The subcomplex corresponding to the boundary of $C_k(N)$ will be denoted by $L_k(N)$. $L_k(N)$ is called the kth level in $A(N)$. It can be unbounded. Note that $C_1(N)$ is precisely the convex polytope surrounding the origin and $L_1(N)$ is its boundary. It is easy to see that a j-face $f \in A(N)$, having level $k-1$, is present in precisely $d-j$ levels $L_k(N), \ldots, L_{k+d-j-1}$. Let us define $M_k(N)$, the kth belt complex, to be the union of $L_{k-1}(N)$, $L_k(N)$ and the d-faces of $C_k(N)$ between $L_{k-1}(N)$ and $L_k(N)$. By convection $C_0(N)$ and $L_0(N)$ are empty complexes. $M_1(N)$ is precisely the convex polytope $C_1(N)$ in $A(N)$ surrounding the origin o. We shall denote the size (i.e., the total number of faces) of any subcomplex Φ of $A(N)$ by $|\Phi|$.

Levels in arrangements become important in connection with Voronoi diagrams because the construction of a kth order Voronoi diagram in R^d can be reduced to the construction of a k-level in R^{d+1} [9]. For this we identify R^d with the coordinate hyperplane $x_{d+1}=0$ in R^{d+1}. We also assume that the origin (basepoint) in R^{d+1}, with respect to which the levels will be defined, is located at $(0, \ldots, 0, -\infty)$. Reference [9] shows that the given collection $\bar{N}$ of sites in $R^d = \{x_{d+1}=0\}$ can be transformed into a set N of hyperplanes in R^{d+1} so that the kth order Voronoi diagram of $\bar{N}$ is obtained by vertically projecting $L_{k+1}(N) \cap L_k(N)$ onto the hyperplane $x_{d+1}=0$ [9].

Henceforth, we shall let z denote a fixed linear function on R^d that is nondegenerate on $A(N)$. This means that the vertices of $A(N)$ have distinct z-coordinates. Almost all linear functions are nondegenerate. So by standard perturbation arguments this assumption can be enforced in our algorithms.

3. Local Minima of a Linear Function on Levels

In this section, we shall prove our bound on the number of local z-minima on levels in arrangements. Let $C_l(N)$ be the l-complex in the arrangment $A(N)$ of hyperplanes. We are interested in knowing how many local minima the given nondegenerate linear function z can attain on $C_l(N)$. Obviously all local z-minima on $C_l(N)$ lie on its boundary $L_l(N)$. We will only be interested in those minima which lie strictly on $L_l(N)$, i.e, on the relative complement $L_l(N) \backslash L_{l-1}(N)$. As it is to be expected, there is a local charachterization of such a minimum: a junction $v \in A(N)$, with level $l-1$, is a local minimum on $L_l(N) \backslash L_{l-1}(N)$, iff v is the (unique) minimum on the convex polytope $\bigcap_{i=1}^{d} H_i^-$, where H_i's are the hyperplanes containing v. Let $w(l)$ denote the number of such minima.

THEOREM 1. $\sum_{l=1}^{k} w(l) = O(k^d)$.

In fact, we shall prove a stronger theorem (which is not needed in the rest of this paper). For this purpose, define, for every real number $s > 0$, $\tau(s, k) = \sum_{l=1}^{k} w(l)/l^s$. Then

THEOREM 2. 1. $\tau(s, k) = O(k^{d-s})$, *for* $s < d$.

2. $\tau(s, k) = O(\log k)$, *for* $s = d$.

3. $\tau(s, k) = O(1)$, *for* $s > d$.

Critical behaviour of $\tau(s, k)$ in the neighbourhood of $s = d$ suggests that $w(l)$ may be $O(k^{d-1+\varepsilon})$, for every $\varepsilon > 0$. It is illuminating to compare Theorem 2 with a similar result for the θ series that was defined in [16] by $\theta(s, k) = \sum_{l=1}^{k} v(l)/l^s$, where $v(l)$ is the number of junctions with level $l-1$ in $A(N)$. It was proved in [16] that:

1. $\theta(s, k) = O(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil - s})$, for $s < \lceil d/2 \rceil$.
2. $\theta(s, k) = O(n^{\lfloor d/2 \rfloor} \log k)$, for $s = \lceil d/2 \rceil$.
3. $\theta(s, k) = O(n^{\lfloor d/2 \rfloor})$, for $s > \lceil d/2 \rceil$.

Note that $v(l)$ is precisely the number of junctions in $L_l(N) \backslash L_{l-1}(N)$. In contrast, $w(l)$ is the number of local z-minima in $L_l(N) \backslash L_{l-1}(N)$.

The proof of Theorem 2, which is probabilistic, is similar to the proof of the above-mentioned result on θ series in [16]. The probabilistic method, pioneered by Erdős [10], has turned out to be quite applicable in computational geometry. See also [5, 7, 12] for other similar applications.

Proof. Given an integer $r > 0$, perform the following experiment. For each hyperplane in the arrangement, independently toss a coin having the probability of success $1/r$. If the toss is successful we retain the hyperplane, otherwise it is discarded. Let ϕ_r be the number of z-minima on the convex polytope containing the origin o in the resulting arrangement. Obviously $\phi_r \leqslant 1$. A fixed local z-minimum $v \in L_l(N) \backslash L_{l-1}(N)$ will be the z-minimum on the convex polytope containing o in the resulting arrangement iff all d hyperplanes containing it are retained and the $(l-1)$ hyperplanes separating it from the origin are discarded. This happens with probability $(1/r)^d (1-1/r)^{l-1}$. It follows that

$$\sum_{l=1}^{n} \left(\frac{1}{r}\right)^d \left(1 - \frac{1}{r}\right)^{l-1} w(l) = E(\phi_r) \leqslant 1. \tag{1}$$

Theorem 1 follows from this equation, if we let $r = k$ and discard the terms on the left with $l > k$. Note that, for $l \leqslant k$, $(1-1/k)^{l-1} \geqslant (1-1/k)^{k-1} \approx 1/e$.

To prove Theorem 2, we rearrange the equation, and obtain, for $s \geqslant 0$,

$$\sum_{l=1}^{n} \frac{1}{r^{s+1}} \left(1 - \frac{1}{r}\right)^{l-1} w(l) = O(r^{d-s-1}). \tag{2}$$

Now sum both sides of Eq. (2) for r ranging from 1 to k', where $k' = ak$ for a suitable constant a to be chosen later. After discarding some terms on the left-hand side, we obtain

$$\sum_{l=1}^{k} \alpha(l)\, w(l) = O\left(\sum_{r=1}^{k'} r^{d-s-1}\right), \tag{3}$$

where

$$\begin{aligned}\alpha(l) &= \sum_{r=1}^{k'} \frac{1}{r^{s+1}}\left(1-\frac{1}{r}\right)^{l-1} \\ &\approx \int_{1}^{k'} \frac{1}{x^{s+1}}\left(1+\frac{1}{x}\right)^{l-1} dx, \\ &= \int_{1/k'}^{1} t^{s-1}(1-t)^{l-1}\, dt \\ &= B(s,l) - \int_{0}^{1/k'} t^{s-1}(1-t)^{l-1}\, dt.\end{aligned}$$

Here $B(s,l) = \int_0^1 t^{s-1}(1-t)^{l-1}\, dt$ denotes the well-known beta function. Using the well-known relation between the beta and the gamma functions, [13],

$$B(s,l) = \frac{\Gamma(s)\,\Gamma(l)}{\Gamma(s+l)} = \frac{1}{s\binom{l-1+s}{s}}, \tag{4}$$

where $\binom{a}{b}$ denotes a generalized binomial coefficient. For $s > 0$, the error term

$$\delta = \int_0^{1/k'} t^{s-1}(1-t)^{l-1}\, dt < \int_0^{1/k'} t^{s-1}\, dt = \left(\frac{1}{k'}\right)^s \frac{1}{s},$$

can be neglected, for every $l \leqslant k$, if $k' = ak$ is chosen to be a sufficiently large multiple of k. Hence, for $l \leqslant k$,

$$\alpha(l) \approx \frac{1}{s\binom{l-1+s}{s}} = \Omega\left(\frac{1}{l^s}\right) \qquad \text{for} \quad s > 0. \tag{5}$$

Now the theorem follows from Eq. (5) and Eq. (3).

4. Linear Programming on Levels

Assume that we are given the $(k-1)$-complex $C_{k-1}(N)$. But we do not know a priori what $C_k(N)$ is. We are also given, as usual, a nondegenerate linear function z. How fast can we determine the $w(k)$ local z-minima in $L_k(N) \backslash L_{k-1}(N)$? For the sake of simplicity, we shall assume in this section that the belt $M_k(N)$ is bounded.

THEOREM 3. *The local z-minima in $L_k(N)\backslash L_{k-1}(N)$ can be determined in $O(w(k)\, n + |L_{k-1}(N)|)$ time.*

Recall that $|L_{k-1}(N)|$ denotes the size of $L_{k-1}(N)$. In the proof of Theorem 3, a certain concept of localization plays a crucial role. As this is quite a general concept, let us define it in the setting of an arbitrary complex. Let Φ be a cell complex. The faces of Φ are assumed to be relatively open and convex. We shall also assume that Φ is a closed cell complex; this means the subset of R^d, obtained by taking the union of all faces in Φ, is closed.

Given a j-face f in Φ, let us define Φ_f, *localization* of Φ w.r.t. f, to be the set of l-faces in Φ, $l \geqslant j$, that are adjacent to f (by convention, f is considered to be adjacent to itself), and this set is assumed to be equipped with the adjacency relationships among the faces in it. Note that we do not consider the faces of dimension $< j$ that are adjacent to f. For $l > j$, the set of l-dimensional faces in Φ_f will be denoted by Φ_f^l. Given a face g adjacent to f, we shall denote its "localization" at f by $[g]_f$; thus $[g]_f \in \Phi_f$. When we denote a face in Φ_f by $[g]_f$, we must remember that $[g]_f$ only tells us the local specification of g at f; it does not tell us everything about the actual face g in Φ. The motivation behind defining localization is that, even if Φ has a complex structure, localizations Φ_f are often simple and easy to determine. This is indeed the case with the complexes $C_k(N)$, $L_k(N)$, and $M_k(N)$. In fact, the localizations at a face f can be readily computed once we know the set of hyperplanes containing f, and the level of f. This follows from the following lemma.

LEMMA 1. *Let f be a j-face in $A(N)$ having level $k-1$, and let $F \subseteq N$ be the hyperplanes containing f. Then, letting $\cong$ denote isomorphism:*

1. $C_{k+i}(N)_f \cong C_i(F)_f$, *for every* $i \geqslant 0$,
2. $L_{k+i}(N)_f \cong L_i(F)_f$, *for every* $i < d-j$, *and*
3. $M_{k+i}(N)_f \cong M_i(F)_f$, *for every* $i \leqslant d-j$.

Proof. Localizations of the levels and belts containing f do not change if we throw away the hyperplanes not containing f. ∎

Another important property of localization is that it is generally possible to do certain local compatibility computations very easily. More precisely, let f be a j-face adjacent to an l-face g, where $l > j$. Assume that we know the local specification of $[g]_f \in \Phi_f$. We are interested in the following computations:

1. Given the local specification of $[h]_g \in \Phi_g$, compute the local specification of $[h]_f \in \Phi_f$.

2. Conversely, given the local specification of $[h]_f \in \Phi_f$, determine if h is adjacent to g, and, if so, compute the local specification of $[h]_g$.

It can be shown that, for the complexes $C_k(N)$, $L_k(N)$, $M_k(N)$, the above computations can be done in $O(1)$ time; the constant depends exponentially on d.

Let us return to the proof Theorem 3. Let f be a d-face in $M_k(N)$. We shall define the floor of f to be $\bar{f} \cap L_{k-1}(N)$, where $\bar{f}$ denotes the closure of f. As it is to be expected, the floor is homeomorphic to a $(d-1)$-ball. To see this, let H be a hyperplane such that every vertex of f is contained in H^-. Then it can be shown that the floor of f is homeomorphic to the "shadow" of f onto H that would be obtained if one were to place a light source at the origin. One can similarly define the roof of a d-face in $M_k(N)$; it is contained in $L_k(N)$. It is easy to show that the floors (and roofs) of two distinct d-faces in $M_k(N)$ have disjoint interiors. In this fashion one obtains a one-to-one correspondence between the d-faces in $M_k(N)$ and their floors (roofs) contained in $L_{k-1}(N)$ (resp. $L_k(N)$).

Given $L_{k-1}(N)$, we shall now show that one can determine the floors of the d-faces in $M_k(N)$ in $O(|L_{k-1}(N)|)$ time. We shall construct a certain graph G on the set of vertices $\bigcup_g M_k(N)_g^d$, where g ranges over the $(d-1)$-faces in $L_{k-1}(N)$. Note that, although we do not a priori know $M_k(N)$, we can certainly compute the localization $M_k(N)_g$, for every face in $L_{k-1}(N)$; this follows from Lemma 1. Edges in G will be determined by the following rule: Let f be a $(d-2)$-face in $L_{k-1}(N)$, and let g_1 and g_2 be two $(d-1)$-faces adjacent to f. We join $[h_1]_{g_1} \in M_k(N)_{g_1}^d$ and $[h_2]_{g_2} \in M_k(N)_{g_2}^d$ by an edge if $[h_1]_f = [h_2]_f$. Figure 1 illustrates this for $d=2$. Edges of G are obtained by applying the above rule at all $(d-2)$-faces in $L_{k-1}(N)$.

LEMMA 2. *The connected components of G are in one-to-one correspondence with the d-faces of $M_k(N)$, and the vertices of G in a connected component corresponding to a d-face, say f, in $M_k(N)$, precisely correspond to the $(d-1)$-faces contained in the floor of f.*

Proof. Easy ∎

By applying Lemma 2, we can determine the set of floors of all d-faces in $M_k(N)$. Actually Lemma 2 gives us only the $(d-1)$-faces in the floor of f and their adjacencies, but from this, one can easily determine the faces of lower dimension contained in the floor, as well as the adjacencies among them.

Fix a connected component F fo G, and let f be the d-face corresponding to it.

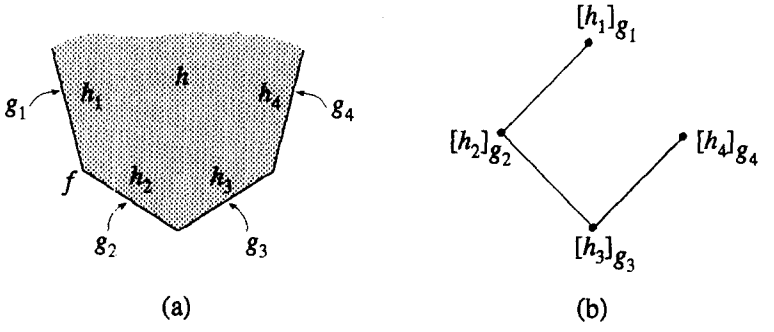

FIG. 1. An example for $d=2$: (a) a two-face h; (b) the corresponding connected component.

One can easily determine if the z-minimum on f lies on its floor, because, given a vectex v contained in the floor of f, one can locally determine, by looking at $[f]_v$ in $M_k(N)_v$, if it is required z-minimum on f. This whole procedure takes time that is linear in the total size of the floor of f. Repeating the procedure for every connected component of G, one can isolate the set S of d-faces (more precisely their floors) in $M_k(N)$ that do not have this property. This means that the restriction of z to $f \in S$ achieves a minimum on the interior of its roof, and thus lies in $L_k(N) \backslash L_{k-1}(N)$. By our assumption $M_k(N)$ is bounded; hence z must achieve a minimum on each face in S. In this fashion, it follows that the local minima in $L_k(N) \backslash L_{k-1}(N)$ that we were interested in finding, are precisely the z-minima on the d-faces in S. As roofs of distinct d-faces in S have disjoint interiors, the cardinality of S is $w(k)$. Now, applying Megiddo's algorithm to each of these d-faces, we can determine the local minima in $L_k(N) \backslash L_{k-1}(N)$ in $O(w(k)\, n)$ time. It is clear that whole procedure of this section takes $O(w(k)\, n + |L_{k-1}(N)|)$ time.

5. Construction of the lth Level $L_l(N)$

In this section we shall show how to construct the lth level $L = L_l(N)$ in the arrangement $A(N)$, given the $(l-1)$th level $L_{l-1}(N)$. As the first level $L_1(N)$ is the convex polytope surrounding the origin ($L_0(N)$ is empty), this will generalize Seidel's result [18]. We shall assume, inductively, that we know the levels of all vertices in $L_{l-1}(N)$. This enables us to determine if a given vertex $v \in L_{l-1}(N)$ belongs to $L_l(N)$ by a pure local computation; see Lemma 1. Our job is to determine the vertices in $L_l(N) \backslash L_{l-1}(N)$, together with the adjacency relationships. We shall assume in this section that L is bounded. Let z be a nondegenerate linear function on R^d as usual. Orient all edges of L in the direction of the increasing z coordinate. Given a vertex $v \in L_l(N) \backslash L_{l-1}(N)$ define the in-degree of v to be the number of edges of $L_l(N)$ that are oriented towards v. We shall prove the following theorem.

Theorem 4. *Suppose that we are given every vertex in $L_l(N) \backslash L_{l-1}(N)$ that has in-degree zero or one. Each such vertex is specified to us by the hyperplanes containing it. Also suppose that each vertex in $L_l(N) \cap L_{l-1}(N)$ is given to us. Then the facial lattice of $L_l(N)$ can be constructed in $O(|L_l(N)| \log n)$ time, where $|L_l(N)|$ is the size of $L_l(N)$.*

The basic idea is to construct $L = L_l(N)$ by scanning it by a hyperplane perpendicular to the z direction. As we shall see below, the vertices of in-degree $\leqslant 1$ in $L_l(N) \backslash L_{l-1}(N)$ cannot be determined easily by scanning and hence they need to be known a priori. Let P^c denote the scanning hyperplane $\{z = c\}$ at "instant" c, and let $P^{\leqslant c}$ denote the closed half space $\{z \leqslant c\}$ to the left of P^c. We shall maintain $L^c = L \cap P^{\leqslant c}$ at every instant c. The *frontier* F^c is defined to be the restriction of L^c to the hyperplane P^c.

Let $v_1, ..., v_m$ be the vertices of L ordered according to the increasing z coordinate. Recall that we are assuming that L is bounded. Let d_j denote the z-coordinate of v_j. Because the function z is nondegenerate, the numbers d_j are distinct. By convention, we shall let $d_0 = -\infty$ and $d_{m+1} = \infty$. The following observations can be made regarding L^c:

1. $L^{d_0} = L^{-\infty} = \varnothing$, and $L^{d_{m+1}} = L^{\infty} = L$.

2. The facial structures of L^c and $L^{c'}$ are the same if c and c' belong to the open interval (d_j, d_{j+1}), where $0 \leqslant j \leqslant m$. As we are only interested in the facial structures, in what follows, we shall treat L^c and $L^{c'}$ as if they were one and the same.

3. For a small enough real number ε, the difference in the facial structures of $L^{d_j - \varepsilon}$ and $L^{d_j + \varepsilon}$ can be determined by simply looking at the localization L^{v_j} of L at v_j.

This observation immediately suggests the following algorithm for constructing the facial structure of L: start with $L^{d_0} = \varnothing$ and move the scanning hyperplane P^c from $c = d_0 = -\infty$ to $c = d_{m+1} = \infty$. During this scanning, we always maintain the facial structure of L^c, where c is the "current" z coordinate of P^c. The facial structure of L^c undergoes a change only when P^c passes through a vertex v_j. This change is completely characterized by the localization $L_{v_j} = L_l(N)_{v_j}$, which can be easily computed, because of Lemma 1. At the end of the scanning proces, we have at our disposal the facial structure of $L_{d_m+1} = M_{\infty} = M$, which is what we sought.

We, of course, do not know a priori the vertices $v_1, ..., v_m$ of L. Fortunately, when the scanning hyperplane is at a z coordinate c, we only need to know that the vertex of L whose z coordinate comes immediately after c. Hence, we shall maintain in the algorithm an *event schedule*, which will be a list of certain vertices in the arrangement $A(N)$, ordered according to their z coordinates. A vertex in the event schedule need not always belong to L. However, the event schedule will satisfy the following important property:

> Whenever the scanning hyperplane is at a z coordinate c, the first vertex in the event schedule will always be the vertex of L whose z coordinate comes immediately after c. (*)

If this property holds, the algorithm given above will still work. So let us see how this property can be forced. As we have already remarked, we can determine which vertices in $L_{l-1}(N)$ belong to $L_l(N)$ by applying Lemma 1. By our assumption, we also know every vertex in $L_l(N) \backslash L_{l-1}(N)$ that has in-degree zero or one. Hence, we can initialize the event schedule by inserting into it all such vertices, and the vertices in $L_l(N) \cap L_{l-1}(N)$. That leaves us with the problem of inserting the vertices in $L_l(N) \backslash L_{l-1}(N)$ with in-degree >1 in the event schedule. This will be done in the course of the algorithm as follows. Consider a vertex $v = v_j$ of in-degree $k > 1$ in $L_l(N) \backslash L_{l-1}(N)$. Then L^c, where $d_{j-1} < c < d_j$, contains k edges, i.e., one-faces,

$e_1, ..., e_k$, which will "meet" in v, if extended in the positive z direction. Let f be two-face of L^c containing two of these edges, say e_1 and e_2. Let $\bar{f}$ be the closure of f. Then notice that the edge $e = \bar{f} \cap P^c$ belongs to the frontier F^c. Moreover, the two edges e_1 and e_2 that meet in v are adjacent to e. This suggests the following invariant on the event schedule. At any given instant c, the event schedule will contain entries for:

1. Every vertex in $L_l(N) \backslash L_{l-1}(N)$, with in-degree zero or one, that lies to the right of P^c, and similarly every vertex in $L_l(N) \cap L_{l-1}(N)$ that lies to the "right" of P^c (i.e., in $\{z > c\}$).

2. Degenerate limits of the edges (one-faces) in the frontier F^c: we say that an edge $e = (u_1, u_2) \in F^c$, having endpoints u_1 and u_2, degenerates at a vertex v, with z-coordinate greater than c, if there are edges $e_1, e_2 \in L^c$, adjacent to u_1 and u_2, respectively, that will meet, if extrapolated forwards, at v. We say v is the degenerate limit of e.

The same vertex can be the degenerate limit of several, but a bounded number of edges in the frontier. A vertex in $L_{l-1}(N) \cap L_l(N)$ can also be a degenerate limit of an edge in the frontier. The degenerate limit of an edge in the frontier F^c always belongs to the arrangement $A(N)$. However, there is no guarantee that it belongs to L. But in such a case the entry for v will be automatically removed before the scanning hyperplane reaches v (see the complete algorithm below.) We shall ensure that, at every instant c, an entry for a vertex v in the event schedule has double pointers to the edges (if any) in the frontier F^c, that degenerate at v.

Now we can a give a high level description of the algorithm. The event schedule will be maintained as a priority queue. It will be initialized in the begining by inserting into it the entries for the vertices $L_l(N) \backslash L_{l-1}(N)$ with in-degree zero or one, and the vertices in $L_l(N) \cap L_{l-1}(N)$.

As long as the event schedule is nonempty, we take the first entry in it, corresponding to a vertex, say v. It will turn out that v always belongs to L. Let a be the z-coordinate of v. Now we proceed as follows.

First, we compute the localization $L_v = L_l(N)_v$, with the help of Lemma 1, in $O(1)$ time. To apply Lemma 1, we need to know the level of v. If $v \in L_{l-1}(N)$ we know this already. Otherwise $v \in L_l(N) \backslash L_{l-1}(N)$, and hence its level is l. The difference in the facial structures of $L^{a-\varepsilon}$ and $L^{a+\varepsilon}$, where ε is infinitisimal, is characterized once we know L_v. However, to carry out the change we need to know where in $L^{a-\varepsilon}$ the change takes place. Towards this end, we shall determine every edge e of $L^{a-\varepsilon}$ adjacent to the frontier $F^{a-\varepsilon}$ that will meet v if extrapolated forwards. If e is adjacent to an edge in the frontier that degenerates at v, then the entry for v in the event schedule will contain a pointer to v. Otherwise, the localization of e at v, which is available to us from L_v, tells use the $d-1$ hyperplanes containing e. If, in the course of the algorithm, we maintain an appropriate dictionary structure on the edges of L^c adjacent to the frontier, the above information uniquely locates e. Once we know all the edges in $L^{a-\varepsilon}$ that meet in v, we know where the change in the facial structure takes place. Now one can routinely

update $L^{a-\varepsilon}$ to $L^{a+\varepsilon}$, making use of the already computed localization L_v. Note that it is important to determine all edges of $L^{a-\varepsilon}$ that meet in v, because $L_{a-\varepsilon}$ need not be connected and some of its pieces can "meet" at v. One special case of interest arises when no edge in $L^{a-\varepsilon}$ meets v. This happens if v is a vertex of in-degree zero in $L_l(N)\backslash L_{l-1}(N)$. In this case $L^{a+\varepsilon}$ can be obtained simply by adding a disjoint simplex, having v as its apex, to $L^{a-\varepsilon}$.

Finally, we remove from the event schedule the degenerate limits of the edges in the frontier $F^{a-\varepsilon}$ that were destroyed during this change. We add to the event schedule the degenerate limit v (if any) of every new edge e in $F^{a+\varepsilon}$, if it was not already in the event schedule. If the event schedule already contained an entry for this limit v (say, because some other edge in the frontier was already known to degenerate at that point) we merely update the entry by incorporating into it a double pointer to e.

This finishes the description of the algorithm. The cost of this algorithm, including the update of the event schedule and the dictionary, is clearly $O(\log n)$ per vertex of L. Hence, the algorithm runs in $O(|L| \log n)$ time.

Let us consider a special case of the above algorithm, when $l = 1$. The first-level $L = L_1(N)$ is just the convex polytope surrounding the origin. In this case, the vertices of in-degree zero or one can be determined in $O(n)$ time per vertex, using Megiddo's algorithm [15]. Hence, the convex polytope L can be constructed in $O(|L| \log n + n^2)$ time, which is precisely Seidel's result [18]. Seidel's algorithm is based on a process, called shelling, that can be associated with any convex hull [3]. Our scanning algorithm for constructing the convex polytope L is just a dual of Seidel's shelling algorithm.

Scanning is one of the earliest known paradigms in computational geometry. Its various applications in the low dimensional problems of computational geometry are well known. It continues to be powerful in higer dimensions as well. The algorithm in this section gives one such application.

6. Algorithm for Constructing $C_k(N)$

Now we are ready to put together various components and give an algorithm for constructing $C_k(N)$. We shall assume now that all hyperplanes in N actually support the convex polytope surrounding the origin. It is crucial to note that this assumption was not required in Section 3 and Section 4. We shall assume that $C_k(N)$ is bounded. This assumption will be removed later.

The outline of our algorithm is very simple: we successively construct $C_1(N), C_2(N), \ldots, C_k(N)$. $C_l(N)$ is constructed from $C_{l-1}(N)$ by adding to it the lth level $L_l(N)$. $L_l(N)$ is constructed by applying Theorem 4. Of course, for this theorem to be applicable, we must determine the vertices of in-degree zero and one in $L_l(N)\backslash L_{l-1}(N)$. The vertices of in-degree zero are precisely local z-minima on $L_l(N)\backslash L_{l-1}(N)$, and hence can be determined by applying the algorithm of Section 4. This leaves us with the task of determining the vertices of in-degree one

in $L_l(N)\backslash L_{l-1}(N)$. Towards that end, we shall first prove that intersection of $C_l(N)$ with any hyperplane P in N gives rise to an l-complex in the $(d-1)$-dimensional arrangement $P \cap A(N)$.

More precisely, let P be any hyperplane in N. By our assumption, P actually supports the convex polytope $C_1(N)$ surrounding the origin o. Let o_P be any point in the interior of the facet of $C_1(N)$ that is supported by P. Let N_P be the set of intersections of the hyperplanes in N with P. Consider the $(d-1)$-dimensional arrangement $A(N_P)$ in P. We shall denote the l-complex of $A(N_P)$ w.r.t. o_P by $C_l(N_P)$. $L_l(N_P)$ and $M_l(N_P)$ are similarly defined.

LEMMA 3. *$C_l(N_P) = P \cap C_l(N)$, $L_l(N_P) = P \cap L_l(N)$ and $M_l(N_P) = P \cap M_l(N)$.*

Proof. It is clear that the exact location of the origin o is immaterial in the definition of $C_l(N)$, $L_l(N)$, and $M_l(N)$, as long as o is located in the interior of the convex polytope $C_1(N)$. Now take o arbitrarily close to o_P. ∎

Suppose v in vertex of in-degree one in $L_l(N)\backslash L_{l-1}(N)$. Let f be the d-cell in $M_l(N)$ containing v. Let P be the hyperplane in N that is spanned by the $d-1$ edges of $L_l(N)$ incident to v, that are oriented away from v. Then it is clear that v is a vertex of in-degree zero on the polytope $f \cap P$, which belongs to $M_l(N_P)$, by Lemma 3. In this fashion, we reduce the problem of determining the vertices of in-degree one in $L_l(N)\backslash L_{l-1}(N)$ to the problem of determining the vertices of in-degree zero in $L_l(N_P)\backslash L_{l-1}(N_P)$, for each $P \in N$. Now we only need to apply the algorithm in Section 4, for each $P \in N$. Note that, because $L_{l-1}(N_P) = P \cap L_{l-1}(N)$, it is readily available to us. Also note that, even if all hyperplanes in N support the polytope $C_1(N)$, all hyperplanes in N_p need not support the convex polytope $C_1(N_P)$. But this causes no problem, because we did not make any such assumption in Section 4.

This finishes the description of the algorithm, assuming that $C_l(N)$ is bounded. Let us now estimate the time taken in the construction of the lth level $L_l(N)$. Determination of the vertices of in-degree zero in $L_l(N)\backslash L_{l-1}(N)$, as in Section 4, takes $O(|L_{l-1}(N)| + w(l)\, n)$ time. Similarly, determination of the vertices of in-degree one in $L_l(N_P)\backslash L_{l-1}(N_P)$, for all $P \in N$, takes $O(\sum_{P \in N} w_P(l)\, n + |L_{l-1}(N_P)|)$ time, where $w_P(l)$ is the number of local z-minima in $L_l(N_P)\backslash L_{l-1}(N_P)$. By Lemma 3, $\sum_{P \in N} |L_{l-1}(N_P)| = O(|L_{l-1}(N)|)$, as each vertex in $L_{l-1}(N)$ belongs to a bounded number of levels of the form $L_{l-1}(N_P)$. Thus the time taken in determining the vertices of in-degree one is $O(|L_{l-1}(N)| + \sum_{P \in N} w_P(l)\, n)$. It now follows from Theorem 4 that $L_l(N)$ can be constructed in additional $O(|L_l(N)| \log n)$ time. Thus the overall time taken in the construction of $L_l(N)$ is $O(w(l) + |L_{l-1}(N)| + |L_l(N)| \log n + \sum_{P \in N} w_P(l))$.

Summing over l and noting that $\sum_{l=1}^{k} |L_l(N)| = O(|C_k(N)|)$, it follows that the construction of $C_k(N)$ takes $O(|C_k(N)| \log n + \sum_{l=1}^{k} w(l)\, n + \sum_{P \in N} \sum_{l=1}^{k} w_P(l)\, n)$ time. By Theorem 2, $\sum_{l=1}^{k} w(l) = O(k^d)$ and $\sum_{l=1}^{k} w_P(l) = O(k^{d-1})$. It follows that the algorithm takes $O(k^{d-1} n^2 + |C_k(N)| \log n)$ time.

It remains to give the required modifications in case $C_l(N)$ is unbounded. Let us add to the set N a set of hyperplanes $x_i = (+ \text{ or } -)\,\delta$, where $i \leqslant d$ and δ is a constant that is so large that the box B defined by these hyperplanes contains all vertices of the arrangement $A(N)$. We can let δ be greater than an easily obtained upper bound on the coordinates of the vertices in $A(N)$. Let N' be the set obtained after adding these hyperplanes to N. Let $C'_l(N')$ denote the subcomplex of $A(N')$ consisting of the faces, with level $\leqslant (l-1)$, that are contained in this box B. Let $L'_l(N')$ be the boundary of $C'_l(N')$, and let $M'_l(N')$ be the subcomplex between $L'_{l-1}(N')$ and $L'_l(N')$.

To construct $C_k(N)$, it suffices to construct $C'_k(N')$. But $C'_k(N')$ is bounded. Hence, the above algorithm, with obvious modifications, will construct $C'_k(N')$. With some minor modifications in the analysis, it can be shown that the running time of this algorithm is also $O(k^{d-1}n^2 + |C_k(N)| \log n)$.

As we have already remarked, the construction of a kth-order Voronoi diagram in R^d can be reduced to the construction a k-level in R^{d+1}. More precisely, the given collection of sites in R^d can be transformed into an arrangement of hyperplanes $A(N)$ in R^{d+1}, with the base point located at $(0, ..., 0, -\infty)$, such that the kth-order Voronoi diagram is obtained by projecting $L_{k+1}(N) \cap L_K(N)$ onto the hyperplane $x_{d+1} = 0$ [9]. It is easy to see that the size of the first k Voronoi diagrams is of the same order of magnitude as $|C_k(N)|$. Moreover, to construct the first k-Voronoi diagrams, it suffices to construct $C_k(N)$. Note that we do not need to construct the $(k+1)$th level $L_{k+1}(N)$ because we can determine if a face in $L_k(N)$ belongs to $L_{k+1}(N) \cap L_k(N)$ by a local computation; see Lemma 1. Indeed, if we constructed $L_{k+1}(N)$, the algorithm will not be output sensitive in the size of the first k Voronoi diagrams, because $|C_{k+1}(N)|$ can have a different order of magnitude than $|C_k(N)|$.

7. Dynamic Algorithm

In this section we give a dynamic algorithm for maintaining Voronoi diagrams of order one to k in R^d. We show how any algorithm for dynamically maintaining first-order Voronoi diagrams can be extended to maintain higher order Voronoi diagrams. In fact, the oracle used by our algorithm is even weaker. The oracle, which will be called an *edge-finder* only needs to provide, during the addition of a new site, one edge of the current *first-order* Voronoi diagram that is conflicting with the new site.

As we have already see, the transformation in [9] reduces the construction of the first k Voronoi diagrams of a set of sites in R^d to the construction of a k-complex $C_k(A)$ in a suitable nonredundant arrangement of hyperplanes in R^{d+1}. Hence, in what follows, we shall only be concerned with the dynamic maintenance of a k-complex $C_k(A)$ in a nonredundant arrangement of hyper planes in R^{d+1}, where A denotes the "current" set of hyperplanes in R^{d+1}. The user is allowed two dynamic update operations. He can add a hyperplane to A. We assume that the

nonredundancy restriction is not violated during the addition. (In the case of Voronoi diagrams it is never violated.) The user can also delete a hyperplane from A.

Our starting point is the randomized, incremental algorithm in [16] for constructing the k-complex in the arrangement formed by the given set N of hyperplanes in R^{d+1}. Given a set of n hyperplanes in R^{d+1}, it can construct the k-complex $C_k(N)$ induced by these hyperplanes in expected $O(k^2n + n \log n)$ time, for $d=2$, and $O(k^{\lceil (d+1)/2 \rceil} n^{\lfloor (d+1)/2 \rfloor})$ time, for $d>2$. It can also be made on-line (semi-dynamic) in a straightforward fashion. Hence, let us first quickly recall it.

The randomized incremental algorithm in [16] simply adds the hyperplanes in N, one at a time, in a random order, so as to obtain a succession of k-complexes $C_k(N^1), C_k(N^2), ..., C_k(N^n) = C_k(N)$, where N^i denotes the set of first i randomly chosen hyperplanes. $C_k(N)$ is the k-complex that we sought. The procedure in [16] for adding the $(i+1)$th randomly chosen hyperplane $S = S_{i+1}$ to $C_k(N^i)$ can be thought of as a two step procedure:

1. Retrieve the edges (i.e., one-faces) of the convex polytope $C_1(N^i) \subseteq C_k(N^i)$ that intersect S.

2. Add S to $C_k(N^i)$; i.e., update $C_k(N^i)$ to $C_k(N^{i+1})$.

The edges in $C_1(N^i)$ intersecting S tell us where to "begin" the addition of S; see [16] for the details. A careful examination of this algorithm reveals that, in fact, it suffices to know just one such intersecting edge. In [16] an edge of $C_1(N^i)$ intersecting S is provided by the so-called "conflict graph." Unfortunately, the conflict graph is an inherently static data structure because it depends on all hyperplanes in N, including the ones that have not been added so far. To make the algorithm on-line, we should be able to determine an intersecting edge in an on-line fashion. This is where we shall use the oracle *edge finder*, which was mentioned earlier. We shall describe this oracle later. Until then we shall simply assume that this oracle provides us an edge of $C_1(N^i)$ intersecting S for free. Once such an edge is determined, the second step in the above addition procedure can be accomplished [16] in time proportional to the resulting structural change ϕ during the addition of S. More precisely, ϕ is defined as the total number of faces (of all dimensions) that are newly created or destroyed during the transition from $C_k(N^i)$ to $C_k(N^{i+1})$. It follows from [16] that the total structural change in the underlying k-complex (in R^{d+1}) over a sequence of n random additions is $O(k^2n)$, for $d=2$, and $O(k^{\lceil (d+1)/2 \rceil} n^{\lfloor (d+1)/2 \rfloor})$, for $d>2$. Thus we obtain

LEMMA 4. *The expected running time of the above on-line algorithm for maintaining a k-complex in R^{d+1} over a sequence of n random additions, ignoring the cost of the edge-finder, is $O(k^2n)$, for $d=2$, and $O(k^{\lceil (d+1)/2 \rceil} n^{\lfloor (d+1)/2 \rfloor})$, for $d>2$. (The cost of the edge finder will turn out to be $O(n \log n)$, for $d=2$, and $O(n^2)$, for $d>2$.)*

Now let us see how to make the above on-line (semi-dynamic) algorithm fully dynamic. Let A denote the "current" set of hyperplanes in R^{d+1} at any given time. Let us see how to delete a hyperplane S from A. Let $A' = A \setminus \{S\}$. Our goal is to obtain $C_k(A')$ from $C_k(A)$ quickly. When $d = 2$, this can be accomplished in time proportional to the structural change that occurs during this transition by using the algorithm in [1]. Next we show that this can be done, for arbitrary d, in time proportional to the structural change, up to a log factor. This will be accomplished by applying a certain modification of the output sensitive scanning algorithm in Section 5 for constructing k-levels. We update $C_k(A)$ to $C_k(A')$ in the following three stages.

1. Construct $L_{k+1}(A) \cap S^+$, where S^+ denotes the *closed* half space bounded by S which does not contain the basepoint o and $L_{k+1}(A)$ denotes the $(k+1)$th level in the arrangement formed by A.

2. "Glue" $L_{k+1}(A) \cap S^+$ on top $C_k(A)$; call the resulting complex $D_k(A)$.

3. Remove $C_k(A) \cap S$.

It can be seen that at the end of the third stage we get $C_k(A')$. The crucial stage above is the construction of $L_{k+1}(A) \cap S^+$. That is where the scanning process in Section 5 can be used. We shall show that $L_{k+1}(A) \cap S^+$ can be constructed in time proportional to its size, up to a log factor. Once $L_{k+1}(A) \cap S^+$ is constructed, the remaining two steps are easy. The whole procedure would then take time proportional to the resulting structural change, as we pass from $C_k(A)$ to $C_k(A')$ (ignoring a log factor).

So let us see how to construct the partial level $L_{k+1} \cap S^+$. The algorithm in Section 5 is for constructing the whole level L_{k+1}, rather than just a part of it, as we are interested in here. It also makes use of a generalization of a linear programming algorithm for fixed dimension (such as Meggido's or Clarkson's algorithm) to k-levels. Although this is fine in the static setting, its use in the current dynamic setting will blow up our algorithm. Here we show how the use of linear programming can be bypassed.

The algorithm in Section 5 constructs $L_{k+1}(A)$ by choosing any linear "non-degenerate" linear functional z, and by scanning $L_{k+1}(A)$ by a moving hyperplane $S^t = \{z = t\}$, starting at $t = -\infty$ and moving to $t = \infty$. Let us choose the functional z such that S coincides with $S^0 = \{z = 0\}$ and the base point o lies in the half space $\{z < 0\}$. Let us imagine scanning $L_{k+1}(A)$ starting at $t = -\infty$. When S^t reaches $S^0 = S$, the algorithm in Section 5 would have constructed $L_{k+1}(A) \cap \{z \leqslant 0\}$. As this part of $L_{k+1}(A)$ is not interesting to us, we ignore what this imaginary algorithm has been doing so far. At $t = 0$, we move from the imaginary domain to the real one, because now the algorithm is about to construct the remaining part $L_{k+1} \cap \{z > 0\} = L_{k+1}(A) \cap S^+$, which is precisely what we are interested in. The following theorem follows by extending the argument in Section 5 in a straightforward fashion.

THEOREM 5. *Orient all edges (one-faces) of $L_{k+1}(A) \cap S^+$ in the increasing z-direction. Define the in-degree of any vertex $v \in (L_{k+1}(A) \setminus L_k(A)) \cap S^+$ as the number of edges of $L_{k+1}(A)$ that are oriented towards v. Assume that we are given all vertices in $(L_{k+1}(A) \setminus L_k(A)) \cap S^+$ with in-degree zero or one and also all vertices in $L_{k+1}(A) \cap L_k(A) \cap S^+$. Then the facial structure of $L_{k+1}(A) \cap S^+$ can be constructed in $O(|L_{k+1}(A) \cap S^+| \log |A|)$ time.*

The vertices in $L_{k+1}(A) \cap L_k(A) \cap S^+$ can be easily determined as follows. Assume, without loss of generality, that $L_{k+1}(A)$ is bounded. This is because we can always restrict the arrangement to a large cube approaching infinity and deal with the intersection of $L_{k+1}(A)$ with this cube. With this assumption, it is easy to see that the one-skeleton of $L_{k+1}(A) \cap L_k(A) \cap S^+$ is connected. Hence, begining with the vertices in $L_{k+1}(A) \cap L_k(A) \cap S$ (which are going to be destroyed eventually), we search within the one-skeleton of $L_k(A)$ so as to visit all vertices in $L_{k+1}(A) \cap L_k(A) \cap S^+$. This search takes $O(|L_{k+1}(A) \cap L_k(A) \cap S^+|)$ time. This easily follows from the following facts: (1) the one-skeleton of $L_k(A)$ is a bounded degree graph, (2) we can locally determine whether the given vertex of $L_k(A)$ belongs to $L_{k+1}(A) \cap L_k(A) \cap S^+$ in $O(1)$ time, (3) the one-skeleton of $L_{k+1}(A) \cap L_k(A) \cap S^+$ is connected.

Now let us turn to the vertices in $(L_{k+1}(A) \setminus L_k(A)) \cap S^+$ of in-degree zero or one. In Section 5 the vertices of in-degree zero or one were determined by a generalization of a linear programming algorithm for fixed dimension to k-levels. In the static setting this was efficient. In the current dynamic setting, however, this would blow up our algorithm. However, when all hyperplanes in A are nonredundant, as in the case of Voronoi diagrams, we go through safely because of the following result.

LEMMA 5. *Every vertex $v \in (L_{k+1}(A) \setminus L_k(A)) \cap S^+$ has in-degree ≥ 2.*

Proof. Case 1. Suppose to the contrary that the in-degree of v is zero. This means that v is a local z-minimum on $(L_{k+1}(A) \setminus L_k(A)) \cap S^+$. Let $C(v)$ be the cone containing the base point o formed by the hyperplanes in A passing through v. It follows that v must be the z-minimum of $C(v)$. But this is impossible because $S = S^0 = \{z = 0\}$ separates v and the base point o and, moreover, $v \in S^+ = \{z > 0\}$.

Case 2. Suppose to contrary that the in-degree of v is one. In this case there are d edges of $L_{k+1}(A) \setminus L_k(A)$ that are oriented away from v. Let $Q \in A$ be the hyperplane in R^{d+1} containing these d edges. Let A_Q be the set of $(d-1)$-dimensional hyperplanes contained in Q that are formed by intersecting the hyperplanes in A with Q. Let $\mathscr{A}(A_Q) = \mathscr{A}(A) \cap Q$ be the arrangement in Q formed by the $(d-1)$-dimensional hyperplanes A_Q. By our nonredundancy assumption, Q supports the convex polytope $L_1(A)$ in $\mathscr{A}(A)$ surrounding the basepoint o. Let o_Q be any point in the interior of $L_1(A) \cap Q$. Define levels $L_k(A_Q)$ in $\mathscr{A}(A_Q)$ with respect to the basepoint o_Q.

Then $L_k(A_Q) = L_k(A) \cap Q$ by Lemma 3. Hence, it follows that v, considered as a vertex in $(L_{k+1}(A_Q) \cap S^+) \setminus (L_k(A_Q) \cap S^+)$ has in-degree zero. Now an argument as in the Case 1 gives a contradition. ▮

The above lemma in conjuction with Theorem 5 implies that the partial level $L_{k+1} \cap S^+$ can be constructed in time proportional to its size (up to a log factor). To summarize:

LEMMA 6. *Deletion of S from $C_k(A)$ can be accomplished in $O(\phi(A, S) \log |A|)$ time, where $\phi(A, S) = \phi_k(A, S)$ denotes the structural change, i.e., the total number of newly created and destroyed faces of all dimensions, as we pass from $C_k(A)$ to $C_k(A')$, and $|A|$ denotes the size of A. When $d = 2$, this can be done in $O(\phi(A, S))$ time.*

Proof. For arbitary d, we have already proved the theorem. When, $d = 2$, one can appropriately use the algorithm in [1] to update $C_k(A)$ to $C_k(A')$. This takes $O(\phi(A, S))$ time. ▮

This finishes the description of our algorithm. Let us now estimate the cost of deleting a random hyperplane S from A. By the above lemma, the cost of deleting S is $O(\phi(A, S) \log |A|)$, for $d > 2$, and $O(\phi(A, S))$, for $d = 2$. Hence, it suffices to estimate the expected value of $\phi(A, S) = \phi_k(A, S)$, when $S \in A$ is randomly chosen. But note that

$$\phi(A, S) = O(|L_{k+1}(A) \cap S^+| + |L_k(A) \cap S^+| + |C_k(A) \cap S|),$$

where S^+ denotes the half bounded by S not containing the basepoint o. But it is easy to see that:

$$\sum_{S \in A} \phi(A, S) = O((k+1)\,|L_{k+1}(A)| + k\,|L_k(A)| + |C_k(A)|).$$

By [5], $|C_k(A)| = O(k^{\lceil (d+1)/2 \rceil}\,|A|^{\lfloor (d+1)/2 \rfloor})$, and the same bound trivially holds for $|L_{k+1}(A)|$ and $|L_k(A)|$ too. This bound for $L_k(A)$ is most probably not tight; however, nothing better is known, for $d > 2$. For $d = 2$, $|L_k(A)|$ (as well as $|L_{k+1}(A)|$) is bounded by $O(k\,|A|)$ [14]. It follows that

$$\sum_{S \in A} \phi_1(A, S) = O(k^{\lceil (d+1)/2 \rceil + 1}\,|A|^{\lfloor (d+1)/2 \rfloor}) \qquad \text{for} \quad d > 2,$$

$$= (k^2\,|A|) \qquad \text{for} \quad d = 2.$$

As every hyperplane S in A is equally likely to be deleted, it follows that the expected cost of a random deletion from A is $O(k^{\lceil (d+1)/2 \rceil - 1} \log |A|)$, for $d < 2$, and $O(k^2)$, for $d = 2$.

The cost of a random addition can be estimated in a similar vein by simply turning the above analysis backwards [17, 19]. Consider the random addition of a new hyperplane S to the current set of hyperplanes A. Let $A' = A \cup \{S\}$. The

structural change, as we pass from $C_k(A)$ to $C_k(A')$, is the same as the structural change, as we pass from $C_k(A')$ to $C_k(A)$, during the *imaginary* deletion of S from A'. Moreover, this imaginary deletion is random; i.e., every $S \in A'$ occurs with the same likelihood, because the addition operation under consideration is random. It follows that the expected structural change during a random addition can be bounded by the same expression as in the case of a random deletion. To summarize:

THEOREM 6. *The expected cost of a random deletion from* $\mathcal{C}_k(A) \subseteq R^{d+1}$ *is* $O(k^{\lceil (d+1)/2 \rceil + 1} |A|^{\lfloor (d+1)/2 \rfloor - 1} \log |A|)$, *for* $d > 2$, *and* $O(k^2)$, *for* $d = 2$. *The expected cost of a random addition to* $C_k(A)$ *is* $O(k^{\lceil (d+1)/2 \rceil + 1} |A|^{\lfloor (d+1)/2 \rfloor - 1})$, *for* $d > 2$, *and* $O(k^2)$, *for* $d = 2$. *This ignores the cost of the edge finder.*

7.1. *Edge Finder*

We now turn to the description of the edge-finder. For the sake of simplicity, we shall restrict ourselves to the k-complexes that arise in connection with higher order Voronoi diagrams, although the whole discussion can be easily generalized to arbitrary k-complexes, as long as the non-redundance assumption is satisfied. Let A, as usual, denote the current set of hyperplanes in R^{d+1}. During the addition of a new hyperplane S, the edge finder is required to provide an edge of $C_1(A)$ intersecting S. First of all, let us remark that it suffices to find an edge of $C_1(A)$ intersecting the half space S^+. Because once such an edge is determined, we can search within the one-skeleton of $C_1(A) \cap S^+$ to determine an edge of $C_1(A)$ intersecting S. The cost of this search is at most $O(|C_1(A) \cap S^+|)$. The latter quantity $|C_1(A) \cap S^+|$ can also be thought of as the structural change $\phi_1(A, s)$ that results during the transition from $C_1(A)$ to $C_1(A')$, $A' = A \cup \{S\}$. As in the previous section, it follows (letting $k = 1$) that the expected value of $\phi_1(A, S)$ during a random addition is $O(|A|^{\lfloor (d+1)/2 \rfloor - 1})$. This cost obviously dominated by the cost in Theorem 6.

Hence, it suffices to find an edge of $C_1(A)$ that intersects the half space S^+. Examination of the transformation in [9] reveals that determining such an edge of $C_1(A)$ is equivalent to determining an edge of the first-order Voronoi diagram, induced by the sites corresponding to A, which is in conflict with (i.e., destroyed during the addition of) the site corresponding to S. In the planar case $d = 2$, any dynamic algorithm for maintaining a first-order Voronoi diagram will provide such an edge. For example, one can use the dynamic algorithm in [17] for this purpose. The expected cost of this algorithm on a random update is $O(\log |A|)$. Thus the expected cost of an edge finder, for $d = 2$, is logarithmic.

When $d > 2$, we can determine an edge of $C_1(A)$ intersecting S^+ very simply as follows: Let $x_1, ..., x_{d+1}$ be the coordinates in R^{d+1}. Let $\bar{A}$ be the sites in correspondence with the hyperplanes in A, where the correspondence is as in [9]. Without loss of generality we can assume that the set $\bar{A}$ of sites is contained in the hyperplane $x_{d+1} = 0$. We can also assume here, without loss of generality, that the basepoint o with respect to which the levels are defined is located at $(0, ..., 0, -\infty)$;

in fact the transformation in [9] already ensures this. In this case the first-order Voronoi diagram of the set of sites $\bar{S}$ is obtained by simply projecting $L_1(A)$ onto the coordinate hyperplane $x_{d+1}=0$.

To accomplish our task, it obviously suffices to find a vertex in $L_1(A)\cap S^+$. Let p be the site in the hyperplane $x_{d+1}=0$ that corresponds to S. A line parallel to the x_{d+1}-axis through p intersects $L_1(A)$ in some point q_0 that is contained in $L_1(A)\cap S^+$ [9]. It is easy to determine the d-face f_0 of $L_1(A)$ containing q_0 in $O(|A|)$ time as $L_1(A)$ has only $|A|$ facets. Choose an arbitrary line L lying in the affine space containing f_0. If we move along L in a direction away from S, we shall meet some $(d-1)$-face f_1 of f_0 in a point q_1 that lies in $L_1(A)\cap S^+$. It is again easy to determine the $(d-1)$-face f_1 containing q_1 in $O(|A|)$ time because the number of $(d-1)$-faces of f_0 is $O(|A|)$. If we inductively proceed in this fashion, we shall eventually determine a vertex $v=q_d$ in $L_1(A)\cap S^+$ (which can possible lie at infinity). Any edge of $L_1(A)$ incident to v will suffice for our purpose. The cost of the above procedure is obviously $O(|A|)$, which is dominated by the cost in Theorem 6, for $d>2$. Combining the discussion in this section with Theorem 6 we obtain:

THEOREM 7. *The expected cost of a random deletion from* $C_k(A)\subseteq R^{d+1}$ *is* $O(k^{\lceil (d+1)/2\rceil+1}|A|^{\lfloor (d+1)/2\rfloor-1}\log|A|)$, *for* $d>2$, *and* $O(k^2+\log|A|)$, *for* $d=2$. *The expected cost of a random addition to* $C_k(A)$ *is* $O(k^{\lceil (d+1)/2\rceil+1}|A|^{\lfloor (d+1)/2\rfloor-1})$, *for* $d>2$, *and* $O(k^2+\log|A|)$, *for* $d=2$.

The relation between Voronoi diagrams in R^d and levels in R^{d+1} implies:

COROLLARY 1. *It is possible to dynamically maintain Voronoi diagrams of order one to k in* R^d *so that: The expected cost of a random deletion from the current set A of sites is* $O(k^{\lceil (d+1)/2\rceil+1}|A|^{\lfloor (d+1)/2\rfloor-1}\log|A|)$, *for* $d>2$, *and* $O(k^2+\log|A|)$, *for* $d=2$. *The expected cost of a random addition to A is* $O(k^{\lceil (d+1)/2\rceil+1}|A|^{\lfloor (d+1)/2\rfloor-1})$, *for* $d>2$, *and* $O(k^2+\log|A|)$, *for* $d=2$.

REFERENCES

1. A. AGARWAL, L. GUIBAS, J. SAXE, AND P. SHOR, A linear time algorithm for computing the Voronoi diagram of a convex polygon, *Discrete Comput. Geom.* **4** (1989), 591–604.
2. J. BOISSONNAT AND M. TEILLAUD, A heirachial representation of objects: The Delaunay tree, *in* "Proceedings of the 2nd ACM Symp. on Comp. Geom, 1986," pp. 260–268.
3. H. BRUGGESSER AND P. MANI, Shellable decompositions of cells and spheres, *Math. Scand.* **29** (1971), 197–205.
4. K. CLARKSON, New applications of random sampling in computational geometry, *Discrete Comput. Geom.* **2** (1987), 195–222.
5. K. CLARKSON AND P. SHOR, Applications of random sampling in computational geometry, II, *Discrete Comput. Geom.* **4** (1989), 387–421.
6. K. CLARKSON, A Las Vegas algorithm for linear programming when the dimension is small, *in* "Proceedings, 29th IEEE Symp. on Foundations of Comput. Sci. 1988," pp. 452–455.
7. K. CLARKSON, H. EDELSBRUNNER, L. GUIBAS, M. SHARIR, AND E. WELZL, Combinatorial complexity bounds for arrangements of curves and surfaces, *in* "Proceedings of the 29th IEEE Symp. on Foundations of Comput. Sci., 1988," pp. 568–579.

8. H. EDELSBRUNNER, Edge skeletons in arrangements with applications, *Algorithmica* **1** (1986), 93–109.
9. H. EDELSBRUNNER AND R. SEIDEL, Voronoi diagrams and arrangements, *Discrete Comput. Geom.* **1** (1986), 25–44.
10. P. ERDŐS AND J. SPENSER, "Probabilistic Methods in Combinatorics," Academic Press/Akademiai Kiado, New York/Budapest, 1974.
11. L. GUIBAS, D. KNUTH, AND M. SHARIR, Randomized incremental construction of Delaunay and Voronoi diagrams, Lect. Notes in Comput. Sci., Vol. 443, pp. 414–431, Springer-Verlag, New York/Berlin, 1990.
12. D. HAUSSLER AND E. WELZL, ε-nets and simplex range querries, *Discrete Comput. Geom.* **2** (1987), 127–151.
13. D. KNUTH, "The Art of Computer Programming," Vol. 2, Addison–Wesley, Reading, MA, 1981.
14. D. LEE, On k-nearest neighbour Voronoi diagrams in the plane, *IEEE Trans. Comput.* **C-31** (1982), 478–487.
15. N. MEGIDDO, Linear programming in linear time when the dimension is fixed, *J. Assoc. Comput. Mach.* **31** (1984), 114–127.
16. K. MULMULEY, On levels in arrangements and Voronoi diagrams, *Discrete Comput. Geom.* **6**, No. 4 (1991), 307–338.
17. K. MULMULEY, Randomized multidimensional search trees: Dynamic sampling, *in* "Proceedings, Annual ACM Symposium on Computational Geometry, June, 1991," pp. 121–131.
18. R. SEIDEL, Constructing higher dimension convex hulls at logarithmic cost per face, *in* "Proceedings, ACM Sympos. on Theory of Computing 1986, Berkely," pp. 404–413.
19. R. SEIDEL, "A Simple and Fast Incremental Randomized Algorithm for Computing Trapezoidal Decompositions and for Traingulating Polygons," Technical Report B 90-07, Freie Universität, Berlin, October 1990.